Kubernetes at UChicago:

PATh, IRIS-HEP Scalable Systems Lab, U.S. ATLAS, and SLATE

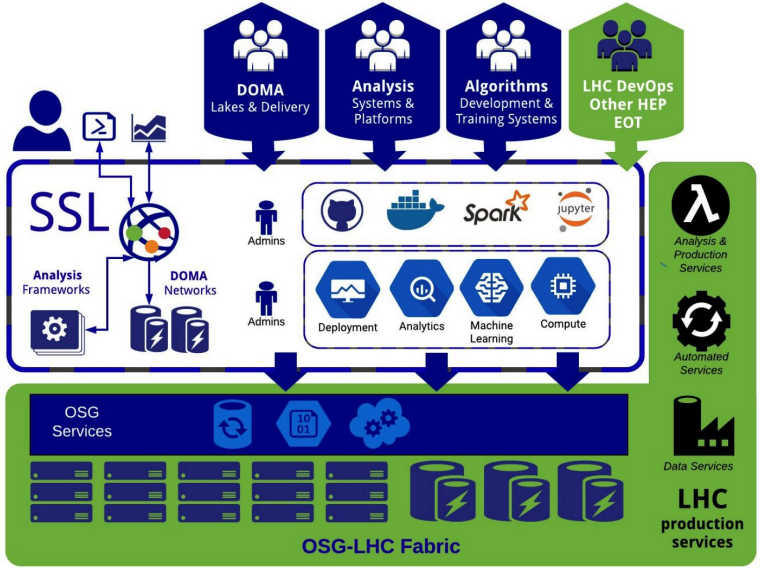
Lincoln Bryant, on behalf of the MANIAC Lab team
University of Chicago
3/16/2022

Motivation

- Kubernetes is a powerful framework for bridging traditional infrastructure and forward-looking tools and platforms
- Have been using Kubernetes (K8S) in various forms at UChicago for over 4 years.
- Foundational for many of our projects!
  - IRIS-HEP Scalable Systems Laboratory
    - R&D facility for advancing HEP software and frameworks
  - U.S. ATLAS Shared Analysis Facility
    - Last-mile interactive analysis with novel platforms.
  - SLATE (Services Layer At The Edge)
    - Federated Application Deployment & Operations
  - MWT2 and service networks for the U.S. ATLAS Computing Facility
  - PATh
    - Hosted Compute Entrypoints and other services
  - FAB
    - FABRIC Across Borders
  - SOTERIA
    - Container registry for open science

IRIS-HEP Scalable Systems Lab

- ServiceX & Coffea-Casa
- Frontier Analytics Platform
- PerfSonar Analytics
- Logstash
- Atlas machine learning platform
- CODAS-HEP training platform
- CMS XCache monitoring
- OSG PATh Hosted CEs
- FuncX



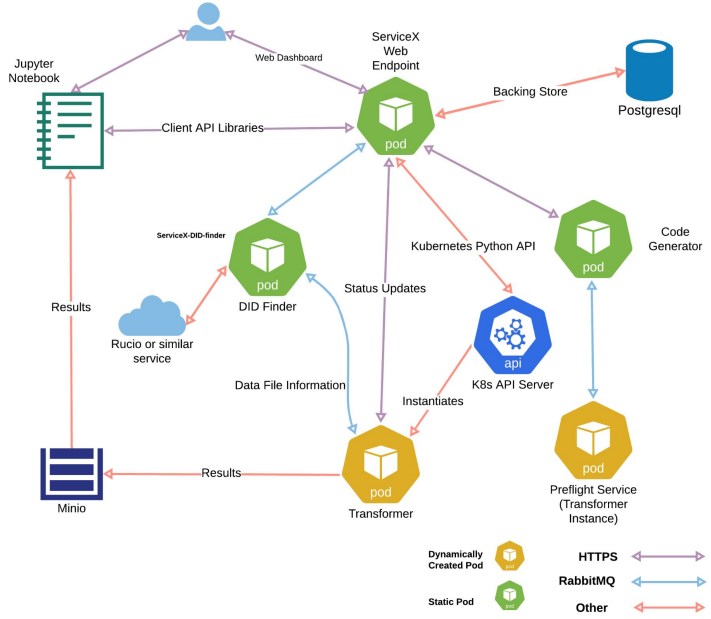
SSL@UC, two K8s clusters: River, River-dev

Analysis Facilities

- We are seeing more and more users reaching for tools that are not a traditional batch interface
  - For example: In the HEP space, a number of interesting new analysis frameworks are being developed, many of which have adopted Kubernetes as an enabling technology
- We have used Kubernetes to build an infrastructure that is flexible enough to support both our batch users and those who want to use novel tools.

One example is ServiceX

- A service that quickly **filters** and **delivers** data in **columnar formats**.
- **Filtering** here means skimming, slimming and augmenting input data. Input data can be xAODs (ATLAS native formats) or flat ROOT files.
- Resulting data can be **delivered** as PyArrow awkward arrays or flat ROOT files.
- Can be used with **Coffea**
- Designed using cloud native software where possible



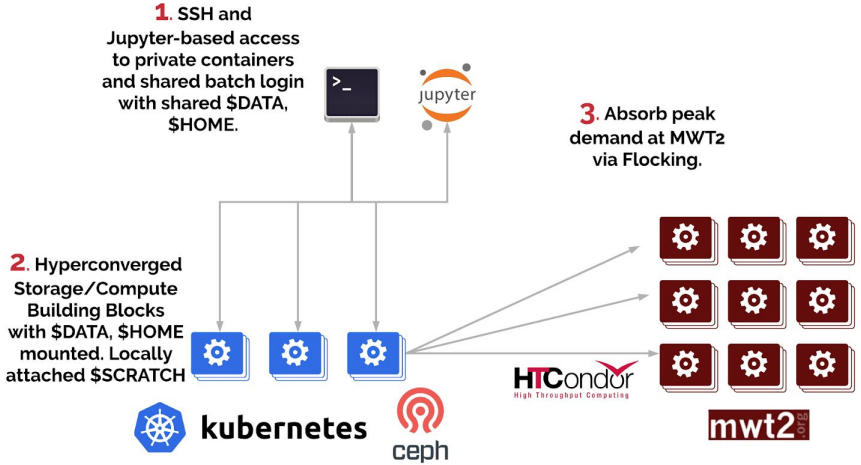
ATLAS Analysis Facility at UChicago

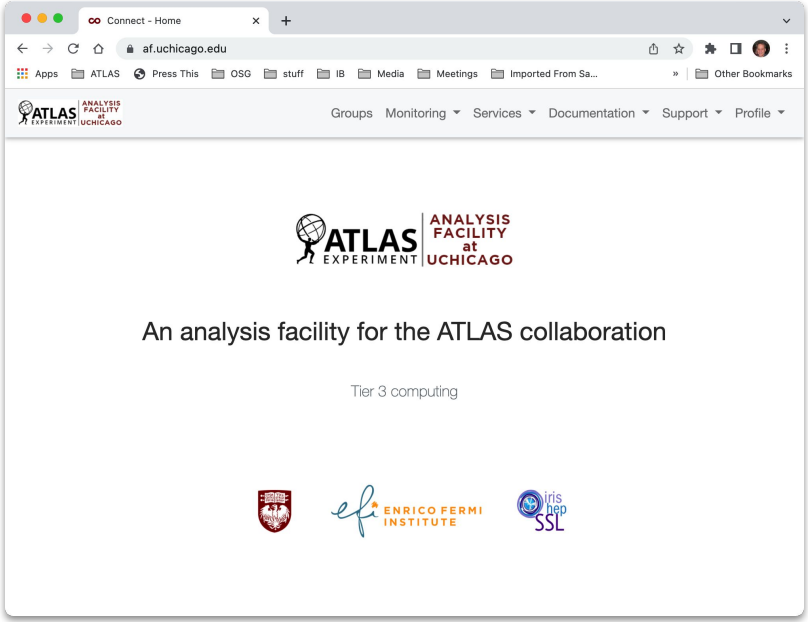
- Charged to build a facility for ATLAS user analysis at the scale of ~1K logical cores and ~1PB storage
- 16 ”hyperconverged” nodes, 6 ”login” nodes, a GPU node, 16 compute nodes with fast local disk, and 25Gbps switching infrastructure
- Hyperconverged nodes for jobs and data.
  - Dual AMD Epyc 7402, 512 GB RAM
  - 16TB HDDs and 1TB NVMes for Ceph storage pool
  - 2TB SSDs for dedicated scratch space for batch
- Login nodes for traditional batch logins plus Jupyter notebooks.
  - Dual AMD Epyc 7402, 256 GB RAM

ATLAS EXPERIMENT | ANALYSIS FACILITY at UCHICAGO

af.uchicago.edu

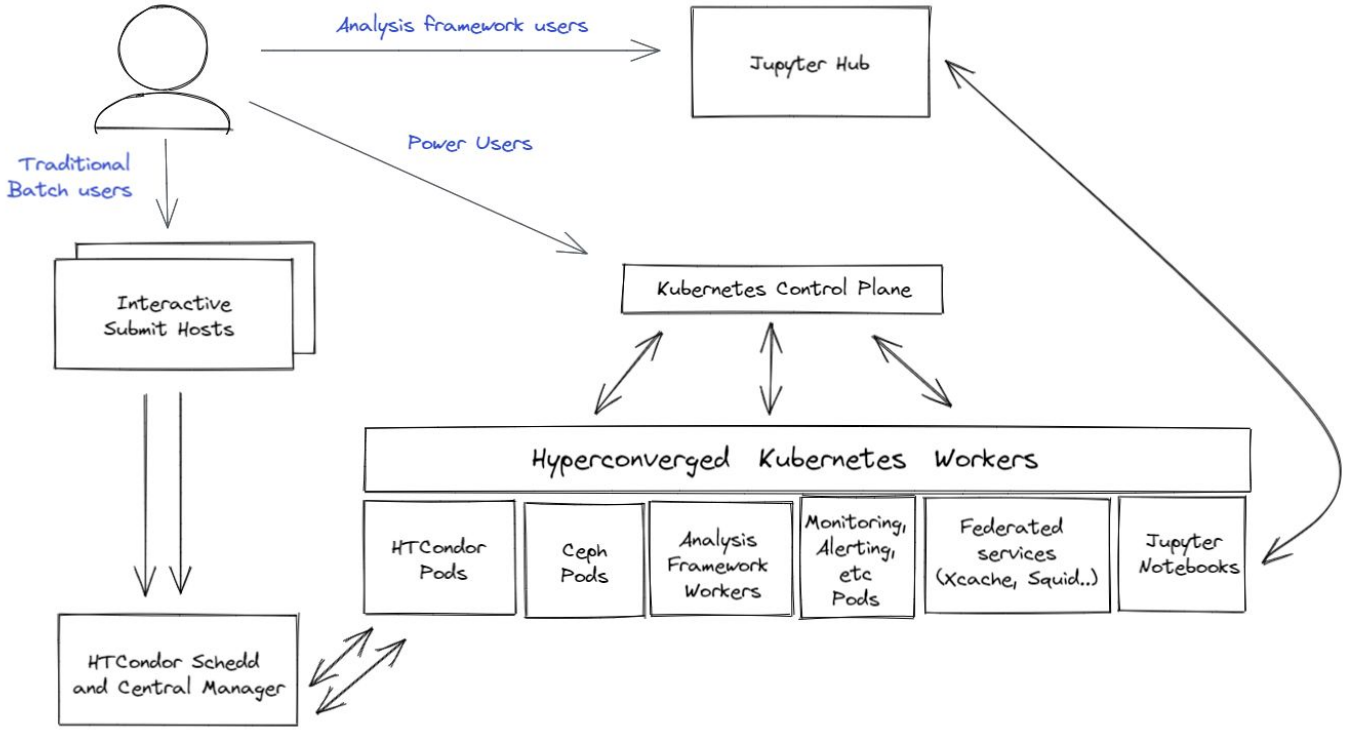
Supporting LHC Run3 and HL-LHC R&D





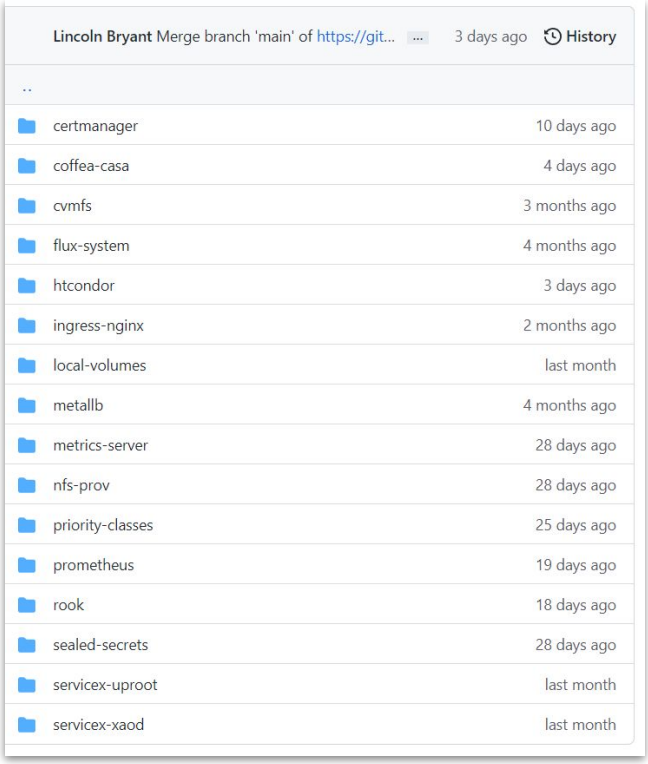
Equipped for Run3 analysis (logins, batch, caches, notebooks) but forward looking with IRIS-HEP services (CoffeaCasa & ServiceX)

Zooming in on the Kubernetes pieces



Tools for declarative deployment – FluxCD

- **Flux CD** – ”GitOps” style application deployment
  - All configuration lives in GitHub, installation/updates/removal all happen via the Flux operator that uses Git as a single source of truth for the cluster.
  - All of the basic Kubernetes extensions are loaded into the Flux repo (Ingress, Load Balancer, monitoring, certificate management, etc)
  - Ceph, HTCondor, etc are also managed by Flux



HTCondor Setup

- Single, unified queue presented to users
  - Any login node, any notebook sees the same queue
- Fully tokenized authentication
  - Each user has a `$HOME/.condor` directory that holds a token allowing job submission to the remote schedd on a shared filesystem
- All execute nodes live in Kubernetes
  - Piecemeal approach to moving daemons into K8S

HTCondor Execute

- Pods configured for 80 logical cores per Worker, partitionable slots
- $HOME, $DATA, CVMFS filesystems mounted into containers
- HTCondor pods are dynamically configured based on values from the Kubernetes downward API, e.g.

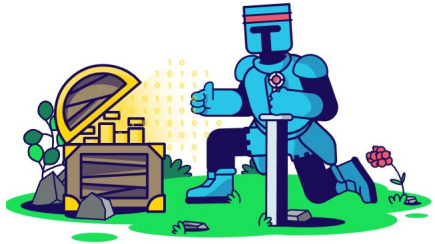
```
resources:
  limits:
    cpu: "84"
    memory: "400G"
    ephemeral-storage: "10G"
  requests:
    cpu: "80"
    memory: "384G"
    ephemeral-storage: "10G"
```

```
- name: _CONDOR_MEMORY
  valueFrom:
  resourceFieldRef:
    containerName: execute
     resource: requests.memory
     divisor: 1Mi
```

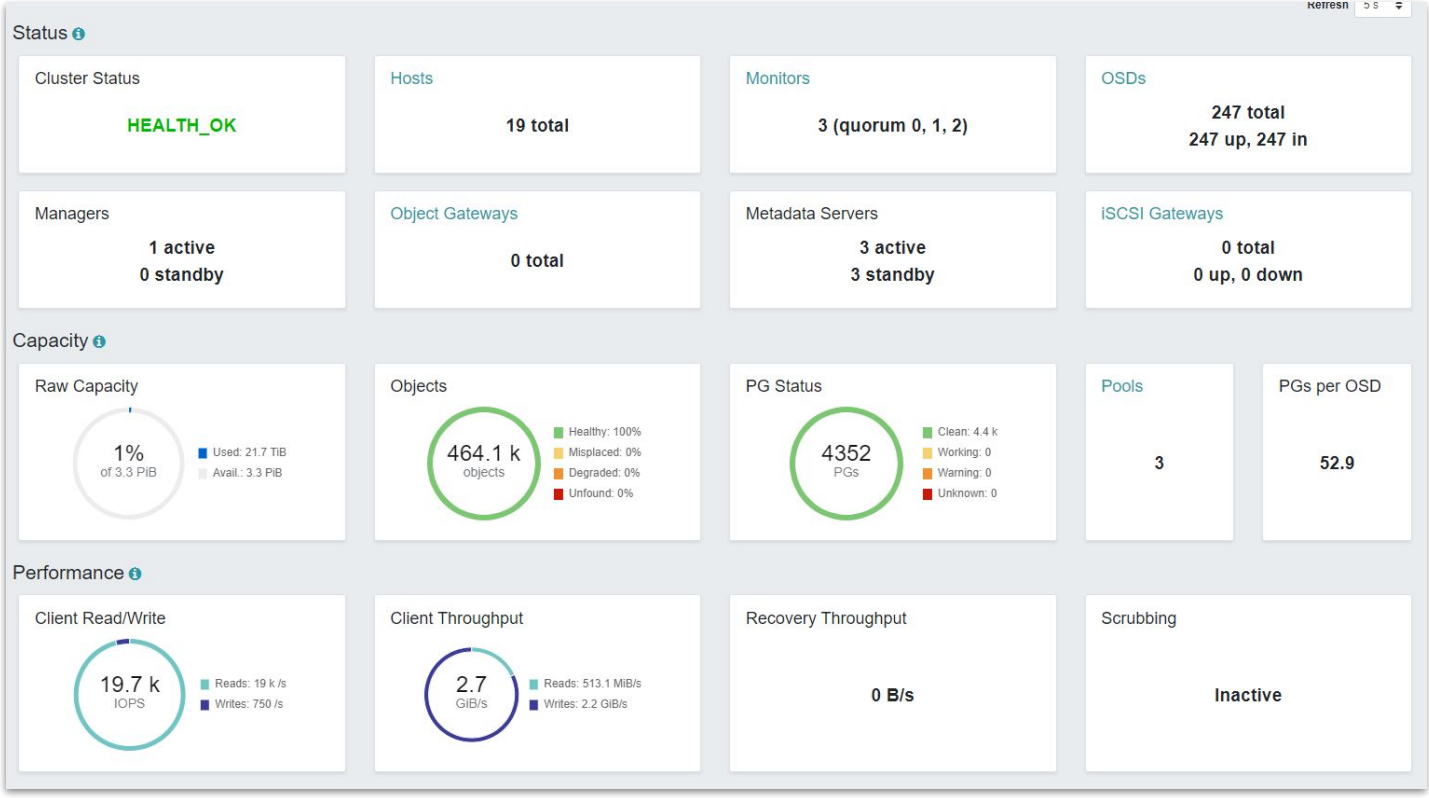
```
$ condor_status slot1@c001 -af Memory
366211
```

Rook & Ceph configuration on the UC AF

- **1PB shared filesystem ($DATA) for users of the AF**
- 228x 16TB HDDs configured for 3x replication
  - Erasure coding is tantalizing for the capacity gains, but we haven't had a good experience with it elsewhere.
- Each node has a dedicated NVMe for Bluestore database (Metadata)
- Each node has a second dedicated NVMe for CephFS
- 3 Active, 3 Standby Metadata Daemons for CephFS
- **Filesystem mountable within Kubernetes and outside.**
- Currently we are not using RADOSGW or RBD.
  - **Focus on performant cluster filesystem for user data.**



Ceph Dashboard



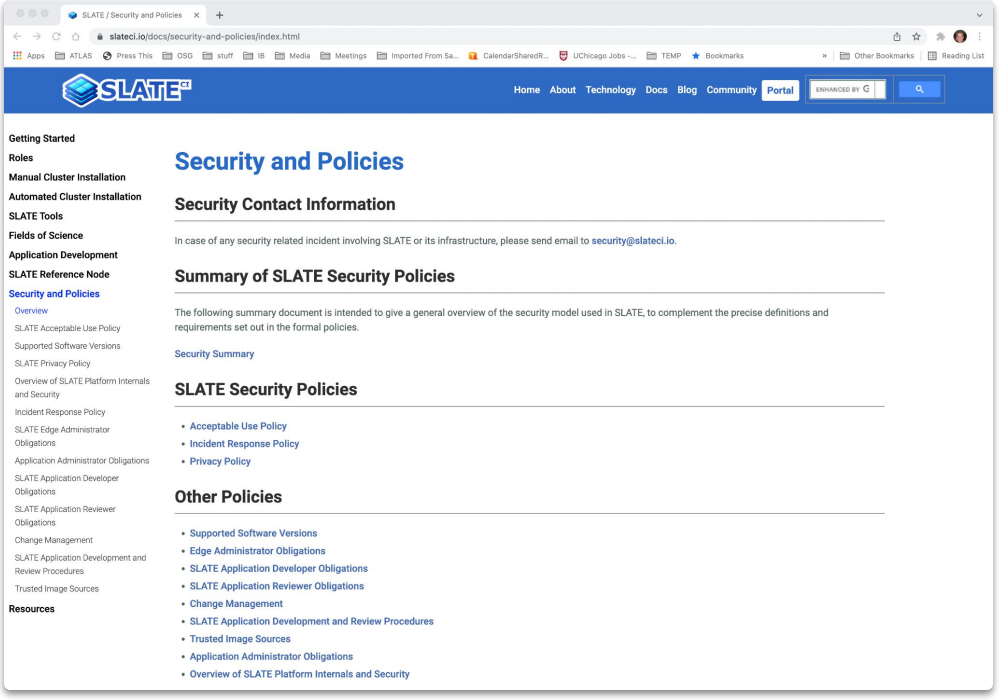
Implementing a federated operation model

- Creating tools and a **trust framework** to create distributed platforms such as CDNs to reduce operational costs and innovate more quickly
- **SLATE** (Services Layer At The Edge) implements distributed service operation and a trust model (close as we can get to a NetFlix model given institutional boundaries)
- **Helm packaged applications**
  - **OSG Entrypoints (both), HTCondor-worker, Frontier Squid, Globus, FTS, XCache, PerfSonar-test, Open OnDemand and more**
  - https://github.com/slateci/slate-catalog
    - usable via Helm even if you don't use SLATE

- SLATE-flavored GitOps
  - Deploy, manage SLATE applications via a single Git repository

Security Policies – with TrustedCI & WLCG

A comprehensive set of security policies that describe needed trust relationships between **application teams**, k8s cluster **admins** and **resource owners**



The US ATLAS Computing Facility has adopted SLATE

- Each Tier 2 in the US setup K8S and installed SLATE for federated service management (5 production clusters)
- Each Tier2 is responsible for keeping the hardware running and the OS and K8S node up to date
- *FedOps* facility team is responsible for keeping the **applications** up to date (**Squid**,**XCache**) on the K8S nodes
- Configured to use SLATE GitOps – single GitHub repository storing all configuration for each application for **each site**.
- SLATE team responsible for ensuring Helm charts and Docker containers are kept reasonably up to date, don't have glaring vulnerabilities, etc.
- Same management interface for Squid and XCache, different teams, partitioned into namespaces by SLATE on the K8S nodes

Evolving our T2 model

Traditional Site

Site admins

Manage Hardware
Install / update OS
Configure and Operate Services
Keep software up to date

Add K8S layer to abstract some of the common pieces at each site

Site under the FedOps Model

Manage Hardware, OS, Kubernetes + SLATE	Site admins
Configure and Operate Services (e.g. Squid, XCache)	FedOps Teams per service
Keep software up to date	SLATE team and trusted container repositories (SOTERIA to curate from trusted sources – CERN, OSG, etc.)

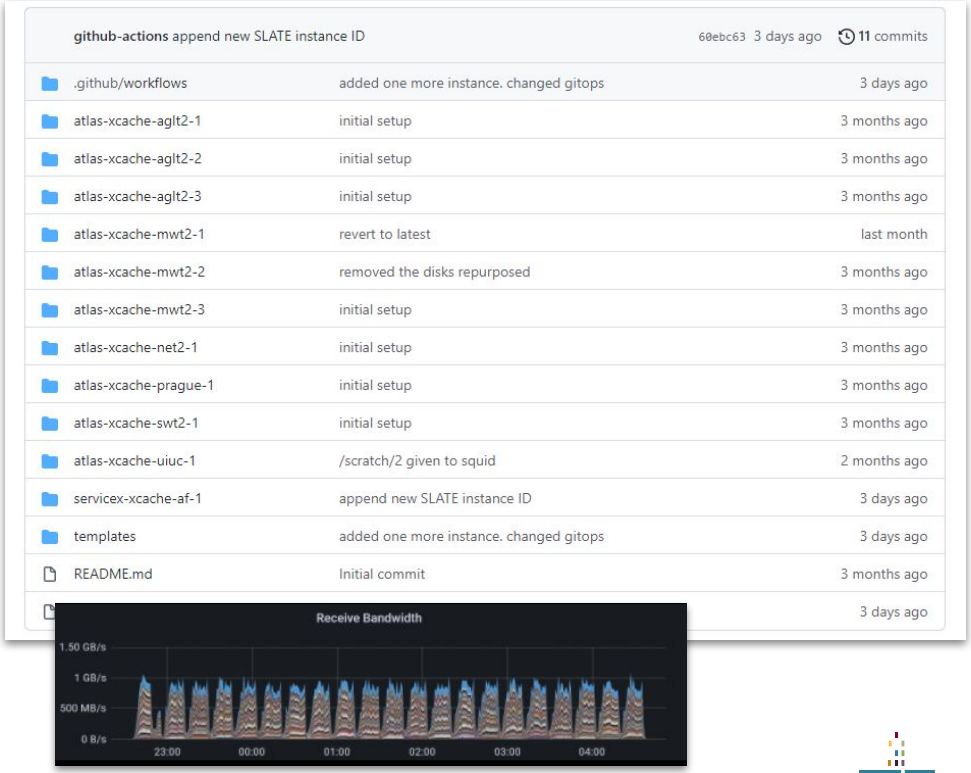
Traditionally, much effort is reproduced at each site and expertise is scattered

Introduce a new layer of abstraction to consolidate expertise, update quickly, iterate on new ideas

XCache Deployments

US ATLAS T2s (UC, IU, UIUC, UM, MSU, UTA, BU), Munich, Prague, Birmingham

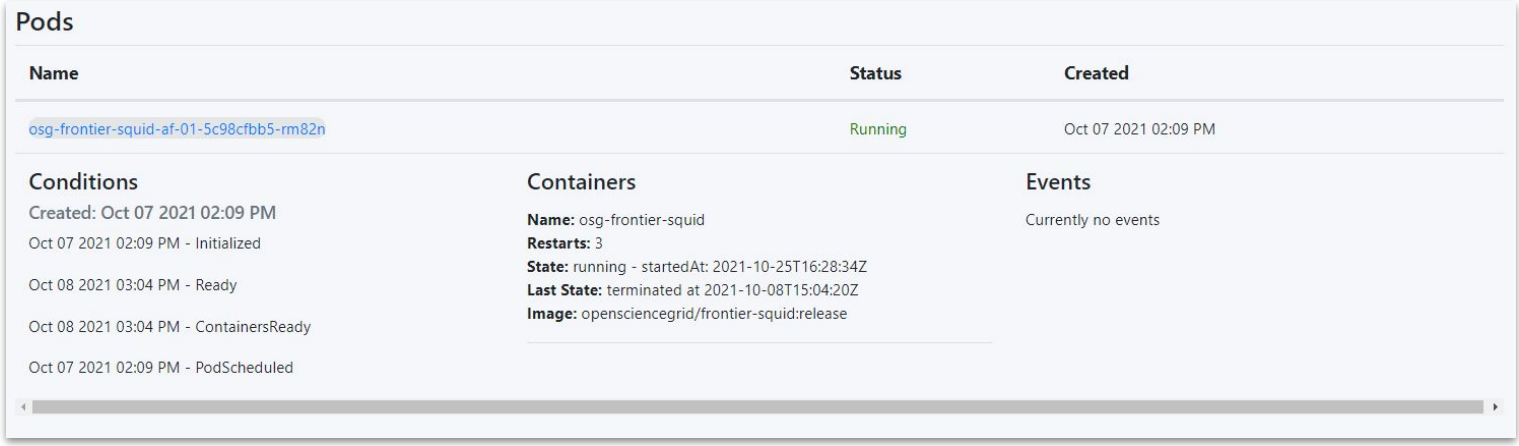
- XRootD-based caching infrastructure
- Optimized access for datasets that aren't geographically nearby
- Deployed via SLATE GitOps to the Analysis Facility
- Hardware
  - 24x3.2TB NVMes
  - 2x25Gbps NIC
  - 2x Xeon Silver 4214, 192GB RAM



github-actions append new SLATE instance ID	60ebc63 3 days ago	11 commits
.github/workflows	added one more instance. changed gitops	3 days ago
atlas-xcache-aglt2-1	initial setup	3 months ago
atlas-xcache-aglt2-2	initial setup	3 months ago
atlas-xcache-aglt2-3	initial setup	3 months ago
atlas-xcache-mwt2-1	revert to latest	last month
atlas-xcache-mwt2-2	removed the disks repurposed	3 months ago
atlas-xcache-mwt2-3	initial setup	3 months ago
atlas-xcache-net2-1	initial setup	3 months ago
atlas-xcache-prague-1	initial setup	3 months ago
atlas-xcache-swt2-1	initial setup	3 months ago
atlas-xcache-uiuc-1	/scratch/2 given to squid	2 months ago
servicex-xcache-af-1	append new SLATE instance ID	3 days ago
templates	added one more instance. changed gitops	3 days ago
README.md	Initial commit	3 months ago
		3 days ago

Squid Deployments

- Software caching, namely Frontier data and CVMFS
- Deployed via SLATE GitOps, managed alongside US Tier 2 squids

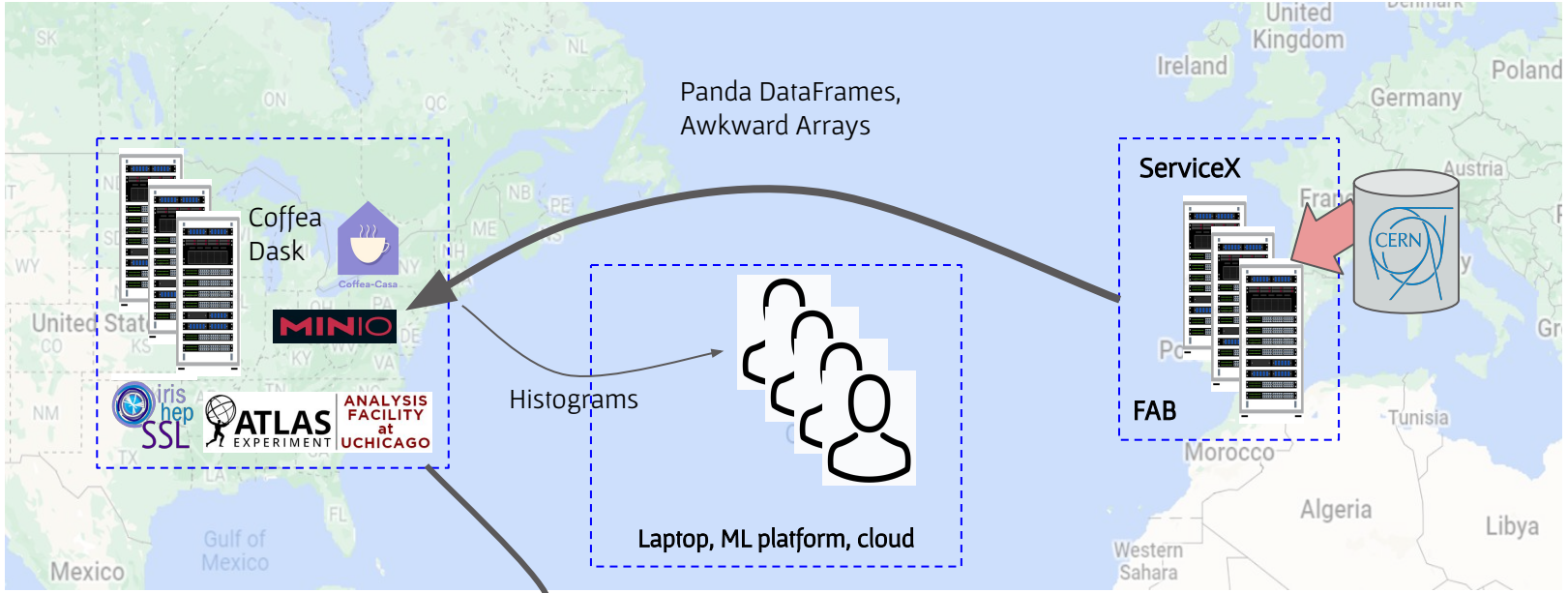


PATh Services

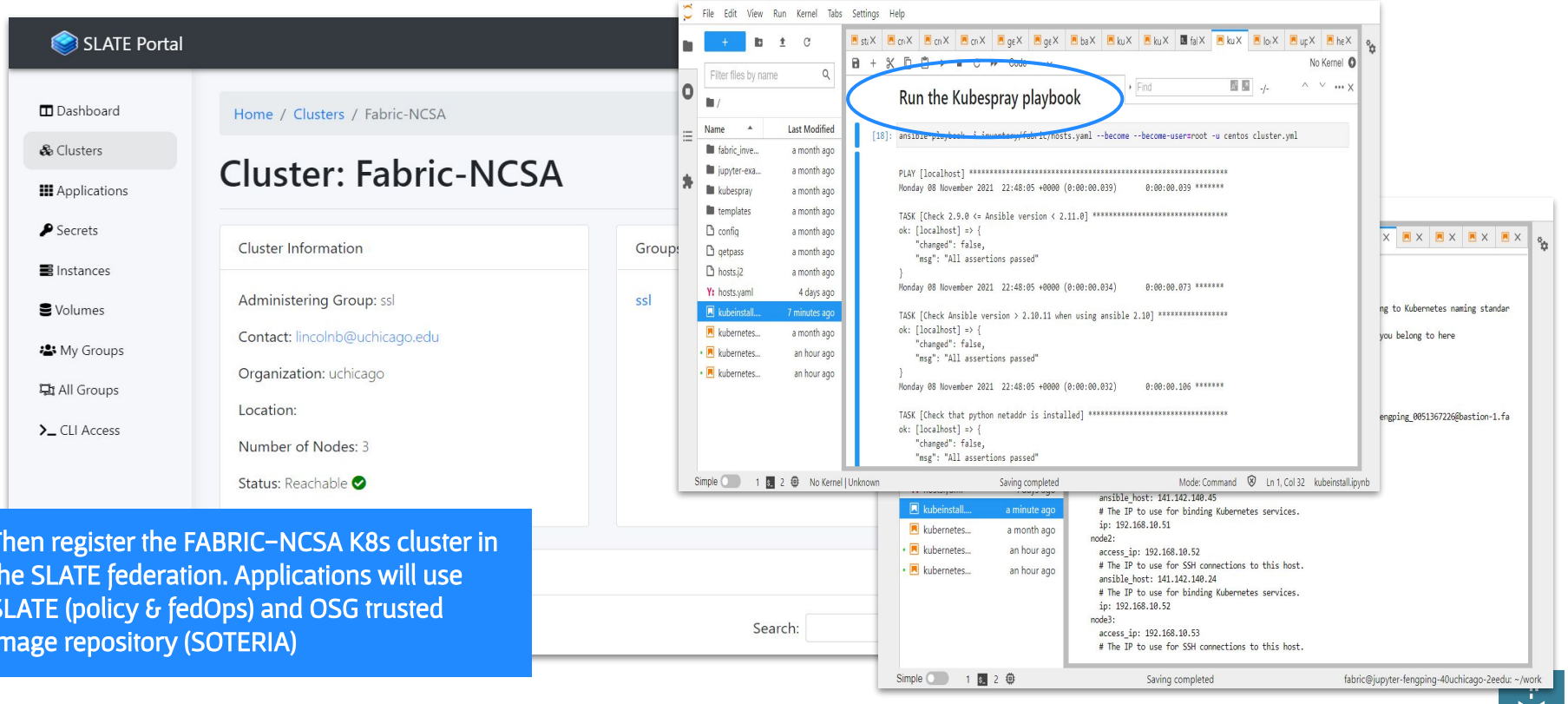
- For PATh, we are taking advantage of the IRIS-HEP SSL River cluster to provide some redundancy and soft load-balancing for applications
  - Hosting many of the Hosted Compute Entrypoints (HostedCEs) for PATh
    - **ASU, FSU, TACC, UIUC, USF, Purdue, AMNH, UCI, UCONN,** and others!
  - Standby of the OSG Harbor container registry service
    - Postgres Operator for database replication
    - Multisite Ceph for object store replication
    - Manual fail over for maintenance or site failure

Deploying into FABRIC

Working with **FAB** (FABRIC Across Borders) to demonstrate ServiceX deployment at CERN, delivery of analysis objects to analysis facilities in the U.S.



Adding the FABRIC k8s to SLATE



Then register the FABRIC–NCSA K8s cluster in the SLATE federation. Applications will use SLATE (policy & fedOps) and OSG trusted image repository (SOTERIA)

CERN->FABRIC->Analysis Facility->Notebook

IRIS-HEP Analysis Grand Challenge Tools Workshop Example

```
[1]: from func_adl_servicex import ServiceXSourceUpROOT
     from hist import Hist
     import awkward as ak
```

We will process only one file from one of the samples. File is accessed using root protocol.

```
[2]: input_files = ['root://eospublic.cern.ch//eos/opendata/atlas/OutreachDatasets/2020-01-22/4lep/MC/mc_345060.ggH125_ZZ4lep.4lep.root']
     treename='mini'
```

CERN data source

The following command does almost everything.

First, it specifies data source by calling ServiceXSourceUpROOT and giving it filepath, root tree containing data, and a name of servicex service to use has to be listed in the file servicex.yaml. In this repo there are two servicex instances that can process this data: uproot-af

Secondly, for every event it gets lepton pT.

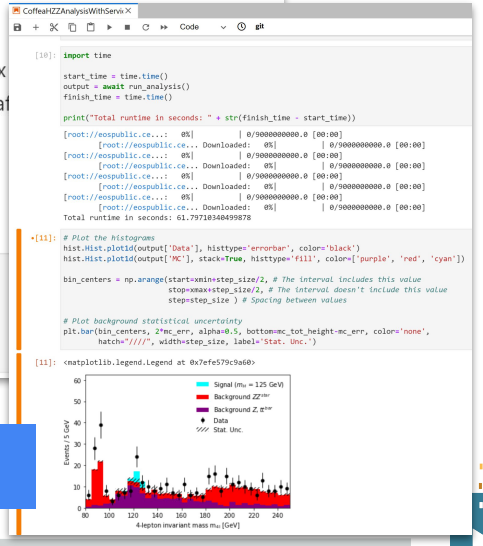
Finally, it specifies that the data should be returned as an Awkward Array.

```
[3]: data = ServiceXSourceUpROOT(input_files, treename, backend_name='uproot-fabric') \
         .Select("lambda e: {'lep_pt': e['lep_pt']}") \
         .AsAwkwardArray() \
         .value()
```

ServiceX on FABRIC-NCSA

Query

Output



All code can be found here.

iris hep SSL

Notebooks on the SSL or AF

Summary

- We have effectively employed K8s as a technology both within our site and in distributed fashion for facilities and collaborations
  - A Kubernetes-based testing platform (Scalable Systems Lab) for IRIS-HEP and others
  - With PATh, declarative operations for Compute Entrypoints
  - A declarative, Kubernetes-based analysis facility for ATLAS that allows traditional batch and forward-looking technologies being developed in IRIS-HEP and other project
  - With SLATE, a **trusted federated operations model** (FedOps) model for deploying, operating, and maintaining services for collaborating computing centers

thank you